

**PORTAL**  
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login  
 Search:  The ACM Digital Library  The Guide  
 +rewrite +cycle +count static file dynamic flash

**THE ACM DIGITAL LIBRARY**

Terms used **rewrite cycle count static file dynamic flash**

Found 1,023 of 154,226

Sort results by  relevance   
 Save results to a Binder  
 Display results  expanded form   
 Search Tips   
 Open results in a new window

Try an Advanced Search  
 Try this search in The ACM Guide

Results 1 - 20 of 200

Result page: 1 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

**1 Services: ELF: an efficient log-structured flash file system for micro sensor nodes**



Hui Dai, Michael Neufeld, Richard Han

November 2004 **Proceedings of the 2nd international conference on Embedded networked sensor systems**

Full text available: pdf(291.68 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

An efficient and reliable file storage system is important to micro sensor nodes so that data can be logged for later asynchronous delivery across a multi-hop wireless sensor network. Designing and implementing such a file system for a sensor node faces various challenges. Sensor nodes are highly resource constrained in terms of limited runtime memory, limited persistent storage, and finite energy. Also, the flash storage medium on sensor nodes differs in a variety of ways from the traditional ...

**Keywords:** eeprom, file system, flash, log structured, reliability, sensor

**2 Enhancing software reliability with speculative threads**



Jeffrey Oplinger, Monica S. Lam

October 2002 **Proceedings of the 10th international conference on Architectural support for programming languages and operating systems**, Volume 37 , 36 , 30 Issue 10 , 5 , 5

Full text available: pdf(1.47 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

This paper advocates the use of a monitor-and-recover programming paradigm to enhance the reliability of software, and proposes an architectural design that allows software and hardware to cooperate in making this paradigm more efficient and easier to program. We propose that programmers write monitoring functions assuming simple sequential execution semantics. Our architecture speeds up the computation by executing the monitoring functions speculatively in parallel with the main computation. For ...

**3 Embra: fast and flexible machine simulation**



Emmett Witchel, Mendel Rosenblum

May 1996 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 24 Issue 1

Full text available: pdf(1.83 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index](#)

of uniprocessors and cache-coherent multiprocessors. When running as part of the SimOS simulation environment, Embra models the processors of a MIPS R3000/R4000 machine faithfully enough to run a commercial operating system and arbitrary user applications. To achieve high simulation speed, Embra uses dynamic binary translation to generate code sequences which simulate the workload. It is the first machine simu ...

#### 4 System support for pervasive applications

Robert Grimm, Janet Davis, Eric Lemar, Adam Macbeth, Steven Swanson, Thomas Anderson, Brian Bershad, Gaetano Borriello, Steven Gribble, David Wetherall  
November 2004 **ACM Transactions on Computer Systems (TOCS)**, Volume 22 Issue 4

Full text available: [pdf\(1.82 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Pervasive computing provides an attractive vision for the future of computing. Computational power will be available everywhere. Mobile and stationary devices will dynamically connect and coordinate to seamlessly help people in accomplishing their tasks. For this vision to become a reality, developers must build applications that constantly adapt to a highly dynamic computing environment. To make the developers' task feasible, we present a system architecture for pervasive computing, called & ...

**Keywords:** Asynchronous events, checkpointing, discovery, logic/operation pattern, migration, one.world, pervasive computing, structured I/O, tuples, ubiquitous computing

#### 5 Continuous profiling: where have all the cycles gone?

Jennifer M. Anderson, Lance M. Berc, Jeffrey Dean, Sanjay Ghemawat, Monika R. Henzinger, Shun-Tak A. Leung, Richard L. Sites, Mark T. Vandevoorde, Carl A. Waldspurger, William E. Weihl

October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles**, Volume 31 Issue 5

Full text available: [pdf\(2.29 MB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

#### 6 Fast detection of communication patterns in distributed executions

Thomas Kunz, Michiel F. H. Seuren

November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

Full text available: [pdf\(4.21 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

#### 7 Session summaries from the 17th symposium on operating systems principle (SOSP'99)

Jay Lepreau, Eric Eide

April 2000 **ACM SIGOPS Operating Systems Review**, Volume 34 Issue 2

Full text available: [pdf\(3.15 MB\)](#) Additional Information: [full citation](#), [index terms](#)

Shun-Tak A. Leung, Richard L. Sites, Mark T. Vandevenne, Carl A. Waldspurger, William E. Weihl

November 1997 **ACM Transactions on Computer Systems (TOCS)**, Volume 15 Issue 4

Full text available: pdf(259.35 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This article describes the Digital Continuous Profiling Infrastructure, a sampling-based profiling system designed to run continuously on production systems. The system supports multiprocessors, works on unmodified executables, and collects profiles for entire systems, including user programs, shared libraries, and the operating system kernel. Samples are collected at a high rate (over 5200 samples/sec. per 333MHz processor), yet with low overhead (1-3% slowdown for most workloads). A ...

**Keywords:** performance understanding, performance-monitoring hardware, profiling, program analysis

**9 Fine-grain access control for distributed shared memory**

Ioannis Schoinas, Babak Falsafi, Alvin R. Lebeck, Steven K. Reinhardt, James R. Larus, David A. Wood

November 1994 **Proceedings of the sixth international conference on Architectural support for programming languages and operating systems**, Volume 29, 28 Issue 11 , 5

Full text available: pdf(1.20 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper discusses implementations of fine-grain memory access control, which selectively restricts reads and writes to cache-block-sized memory regions. Fine-grain access control forms the basis of efficient cache-coherent shared memory. This paper focuses on low-cost implementations that require little or no additional hardware. These techniques permit efficient implementation of shared memory on a wide range of parallel systems, thereby providing shared-memory codes with a portability ...

**10 ESP: a language for programmable devices**

Sanjeev Kumar, Yitzhak Mandelbaum, Xiang Yu, Kai Li

May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5

Full text available: pdf(1.60 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper presents the design and implementation of Event-driven State-machines Programming (ESP)—a language for programmable devices. In traditional languages, like C, using event-driven state-machine forces a tradeoff that requires giving up ease of development and reliability to achieve high performance. ESP is designed to provide all of these three properties simultaneously.

ESP provides a comprehensive set of features to support development of compact and modular programs. ...

**11 Queue management: Persistent dropping: an efficient control of traffic aggregates**

Hani Jamjoom, Kang G. Shin

August 2003 **Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications**

Full text available: pdf(804.16 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

to the point of service failure. Traditional rate-based drop policies, such as Random Early Drop (RED), become ineffective in such situations since clients tend to be persistent, in the sense that they make multiple retransmission attempts before aborting their connection. As it is built into TCP's congestion control ...

**Keywords:** flash crowd events, modeling, optimization, queue management

**12 Multithreading and value prediction: Speculative lock elision: enabling highly concurrent multithreaded execution**

Ravi Rajwar, James R. Goodman

December 2001 **Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture**

Full text available:

pdf(1.37 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)  
[Publisher Site](#)

Serialization of threads due to critical sections is a fundamental bottleneck to achieving high performance in multithreaded programs. Dynamically, such serialization may be unnecessary because these critical sections could have safely executed concurrently without locks. Current processors cannot fully exploit such parallelism because they do not have mechanisms to dynamically detect such false inter-thread dependences. We propose *Speculative Lock Elision (SLE)*, a novel micro-architecture ...

**13 Reading text from computer screens**

Carol Bergfeld Mills, Linda J. Weldon

December 1987 **ACM Computing Surveys (CSUR)**, Volume 19 Issue 4

Full text available:

pdf(3.33 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

This paper reviews empirical studies concerning the readability of text from computer screens. The review focuses on the form and physical attributes of complex, realistic displays of text material. Most studies comparing paper and computer screen readability show that screens are less readable than paper. There are many factors that could affect the readability of computer screens. The factors explored in this review are the features of characters, the formatting of the screen, the contrast ...

**14 Experience with a software-defined machine architecture**

David W. Wall

May 1992 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 14 Issue 3

Full text available:

pdf(2.86 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

We have built a system in which the compiler back end and the linker work together to present an abstract machine at a considerably higher level than the actual machine. The intermediate language translated by the back end is the target language of all high-level compilers and is also the only assembly language generally available. This lets us do intermodule register allocation, which would be harder if some of the code in the program had come from a traditional assembler, out of sight of ...

**Keywords:** RISC, graph coloring, intermediate language, interprocedural, optimization, pipeline scheduling, profiling, register allocation, register windows

Vanderbilt, P. G. Kjeldsberg

April 2001 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**,  
Volume 6 Issue 2

Full text available:  pdf(339.91 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

We present a survey of the state-of-the-art techniques used in performing data and memory-related optimizations in embedded systems. The optimizations are targeted directly or indirectly at the memory subsystem, and impact one or more out of three important cost metrics: area, performance, and power dissipation of the resulting implementation. We first examine architecture-independent optimizations in the form of code transformations. We next cover a broad spectrum of optimizati ...

**Keywords:** DRAM, SRAM, address generation, allocation, architecture exploration, code transformation, data cache, data optimization, high-level synthesis, memory architecture customization, memory power dissipation, register file, size estimation, survey

**16 Specialization tools and techniques for systematic optimization of system software** 

Dylan McNamee, Jonathan Walpole, Calton Pu, Crispin Cowan, Charles Krasic, Ashvin Goel, Perry Wagle, Charles Consel, Gilles Muller, Renaud Marlet

May 2001 **ACM Transactions on Computer Systems (TOCS)**, Volume 19 Issue 2

Full text available:  pdf(178.52 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Specialization has been recognized as a powerful technique for optimizing operating systems. However, specialization has not been broadly applied beyond the research community because current techniques based on manual specialization, are time-consuming and error-prone. The goal of the work described in this paper is to help operating system tuners perform specialization more easily. We have built a specialization toolkit that assists the major tasks of specializing operating systems. We de ...

**Keywords:** operating system specialization, optimization, software architecture

**17 On compile-time query optimization in deductive databases by means of static filtering** 

Michael Kifer, Eliezer L. Lozinski

September 1990 **ACM Transactions on Database Systems (TODS)**, Volume 15 Issue 3

Full text available:  pdf(3.49 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

We extend the query optimization techniques known as algebraic manipulations with relational expressions [48] to work with deductive databases. In particular, we propose a method for moving data-independent selections and projections into recursive axioms, which extends all other known techniques for performing that task [2, 3, 9, 18, 20]. We also show that, in a well-defined sense, our algorithm is optimal among the algorithms that propagate data-independent selections through recursion.

**Keywords:** dataflow, deductive databases, filtering, fixpoint, graph representation, inference, projection, recursive rules, selection

**18 Data abstraction and information hiding** 

K. Rustan M. Leino, Greg Nelson

Full text available: pdf(469.27 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This article describes an approach for verifying programs in the presence of data abstraction and information hiding, which are key features of modern programming languages with objects and modules. This article draws on our experience building and using an automatic program checker, and focuses on the property of *modular soundness*: that is, the property that the separate verifications of the individual modules of a program suffice to ensure the correctness of the composite program. We fo ...

**Keywords:** Abstract variables, abstraction dependencies, extended static checking, modifies clauses, modular verification, object-oriented programming, specifications

#### **19 Special issue on persistent object systems: Orthogonally persistent object systems**

Malcolm Atkinson, Ronald Morrison

July 1995 **The VLDB Journal — The International Journal on Very Large Data Bases,**  
Volume 4 Issue 3

Full text available: pdf(5.02 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

Persistent Application Systems (PASs) are of increasing social and economic importance. They have the potential to be long-lived, concurrently accessed, and consist of large bodies of data and programs. Typical examples of PASs are CAD/CAM systems, office automation, CASE tools, software engineering environments, and patient-care support systems in hospitals. Orthogonally persistent object systems are intended to provide improved support for the design, construction, maintenance, and operation o ...

**Keywords:** database programming languages, orthogonal persistence, persistent application systems, persistent programming languages

#### **20 Exploiting dead value information**

Milo M. Martin, Amir Roth, Charles N. Fischer

December 1997 **Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.38 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)  
[Publisher Site](#)

We describe Dead Value Information (DVI) and introduce three new optimizations which exploit it. DVI provides assertions that certain register values are dead, meaning they will not be read before being overwritten. The processor can use DVI to track dead registers and dynamically eliminate unnecessary save and restore instructions from the execution stream at procedure calls and context switches. Our results indicate that dynamic saves and restore instances can be reduced by 46% for procedure c ...

Results 1 - 20 of 200

Result page: **1** [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.  
[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

**PORTAL**  
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login  
 Search:  The ACM Digital Library  The Guide  
 +re-write +cycle +count static file dynamic

**THE ACM DIGITAL LIBRARY** [Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Terms used **re write cycle count static file dynamic**

Found **62 of 154,226**

Sort results by **relevance**   
 Display results **expanded form**   
 Save results to a Binder  
 Search Tips  
 Open results in a new window

[Try an Advanced Search](#)  
[Try this search in The ACM Guide](#)

Results 1 - 20 of 62

Result page: **1 2 3 4 next**

Relevance scale

- 1 Register allocation across procedure and module boundaries**  
 Vatsa Santhanam, Daryl Odnet  
 June 1990 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1990 conference on Programming language design and implementation**, Volume 25 Issue 6  
 Full text available: [pdf\(1.51 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper describes a method for compiling programs using interprocedural register allocation. A strategy for handling programs built from multiple modules is presented, as well as algorithms for global variable promotion and register spill code motion. These algorithms attempt to address some of the shortcomings of previous interprocedural register allocation strategies. Results are given for an implementation on a single register file RISC-based architecture ...

- 2 Optimising hot paths in a dynamic binary translator**  
 David Ung, Cristina Cifuentes  
 March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1  
 Full text available: [pdf\(890.10 KB\)](#) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

In dynamic binary translation, code is translated "on the fly" at run-time, while the user perceives ordinary execution of the program on the target machine. Code fragments that are frequently executed follow the same sequence of flow control over a period of time. These fragments form a hot path and are optimised to improve the overall performance of the program. Multiple hot paths may also exist in programs. A program may choose to execute in one hot path for some time, but later switch to another ...

**Keywords:** binary translation, dynamic compilation, dynamic execution, run-time profiling

- 3 Energy and Delay Considerations: Unified architecture level energy-efficiency metric**  
 Victor Zyuban  
 April 2002 **Proceedings of the 12th ACM Great Lakes symposium on VLSI**  
 Full text available: [pdf\(109.68 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

The development of power-efficient microprocessors presents the need to consider power consumption at early stages of design, particularly at the system level. This paper presents a unified architecture level energy-efficiency metric that can be used to evaluate the power efficiency of various system designs.

level stages, and the opportunity for making power-performance tradeoffs is the largest. Design modifications to the ISA and microarchitecture, however, affect most (if not all) parameters of the design, including architectural speed, co ...

**Keywords:** architecture, energy, energy-efficiency, metric, microarchitecture, performance, power

#### **4 System support for automatic profiling and optimization**

Xiaolan Zhang, Zheng Wang, Nicholas Gloy, J. Bradley Chen, Michael D. Smith  
October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles**, Volume 31 Issue 5

Full text available:  pdf(1.95 MB) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

#### **5 Building a robust software-based router using network processors**

Tammo Spalink, Scott Karlin, Larry Peterson, Yitzchak Gottlieb  
October 2001 **ACM SIGOPS Operating Systems Review , Proceedings of the eighteenth ACM symposium on Operating systems principles**, Volume 35 Issue 5

Full text available:  pdf(1.49 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Recent efforts to add new services to the Internet have increased interest in software-based routers that are easy to extend and evolve. This paper describes our experiences using emerging network processors---in particular, the Intel IXP1200---to implement a router. We show it is possible to combine an IXP1200 development board and a PC to build an inexpensive router that forwards minimum-sized packets at a rate of 3.47Mpps. This is nearly an order of magnitude faster than existing pure PC-base ...

#### **6 Server performance and scalability: A method for transparent admission control and request scheduling in e-commerce web sites**

Sameh Elnikety, Erlich Nahum, John Tracey, Willy Zwaenepoel  
May 2004 **Proceedings of the 13th international conference on World Wide Web**

Full text available:  pdf(179.28 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

This paper presents a method for admission control and request scheduling for multiply-tiered e-commerce Web sites, achieving both stable behavior during overload and improved response times. Our method externally observes execution costs of requests online, distinguishing different request types, and performs overload protection and preferential scheduling using relatively simple measurements and a straight forward control mechanism. Unlike previous proposals, which require extensive changes to ...

**Keywords:** admission control, dynamic web content, load control, request scheduling, web servers

#### **7 Memory simulators and software generators**

Guillermo Jiménez-Pérez, Don Batory  
May 1997 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 1997 symposium on Software reusability**, Volume 22 Issue 3

Full text available:  pdf(1.30 MB) Additional Information: [full citation](#), [references](#), [index terms](#)

Jalme H. Moreno, Mayan Moudgil

July 1997 **Proceedings of the 11th international conference on Supercomputing**

Full text available:  pdf(1.06 MB) Additional Information: [full citation](#), [references](#), [index terms](#)

**9 Astrophysical N-body simulations using hierarchical tree data structures**

M. S. Warren, J. K. Salmon

December 1992 **Proceedings of the 1992 ACM/IEEE conference on Supercomputing**

Full text available:  pdf(709.46 KB) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

**10 Processors and accelerators for embedded applications: The iCOREtm 520 MHz synthesizable CPU core**

Nick Richardson, Lun Bin Huang, Razak Hossain, Tommy Zounes, Naresh Soni, Julian Lewis  
June 2002 **Proceedings of the 39th conference on Design automation**

Full text available:  pdf(360.25 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

This paper describes a new implementation of the ST20-C2 CPU architecture. The design involves an eight-stage pipeline with hardware support to execute up to three instructions in a cycle. Branch prediction is based on a 2-bit predictor scheme with a 1024-entry Branch History Table and a 64 entry Branch Target Buffer and a 4-entry Return Stack. The implementation of all blocks in the processor was based on synthesized logic generation and automatic place and route. The full design of the CPU fro ...

**Keywords:** ASIC, CPU, branch-prediction, cache, embedded, high-frequency, microarchitecture, pipeline, st20, synthesis

**11 Compiler-directed selection of dynamic memory layouts**

Mahmut Kandemir, Ismail Kadayif

April 2001. **Proceedings of the ninth international symposium on Hardware/software codesign**

Full text available:  pdf(650.29 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Compiler technology is becoming a key component in the design of embedded systems, mostly due to increasing participation of software in the design process. Meeting system-level objectives usually requires flexible and retargetable compiler optimizations that can be ported across a wide variety of architectures. In particular, source-level compiler optimizations aiming at increasing locality of data accesses are expected to improve the quality of the generated code. Previous compiler-based ap ...

**Keywords:** array reuse, data dependence, data locality, memory layout optimization, software compilation

**12 Dynamically Trading Frequency for Complexity in a GALS Microprocessor**

Steven Dropsho, Greg Semeraro, David H. Albonesi, Grigoris Magklis, Michael L. Scott

December 2004 **Proceedings of the 37th International Symposium on Microarchitecture**

Full text available:  pdf(209.21 KB) Additional Information: [full citation](#), [abstract](#)

Microprocessors are traditionally designed to provide "best overall" performance across a wide range of applications and operating environments. Several groups have proposed

saving approach: dividing the processor into domains and dynamically changing the clock frequency and voltage within each domain during phases when the full ...

### **13 WaveScalar**

Steven Swanson, Ken Michelson, Andrew Schwerin, Mark Oskin

December 2003 **Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture**

Full text available: [pdf\(228.98 KB\)](#) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

Silicon technology will continue to provide an exponential increase in the availability of raw transistors. Effectively translating this resource into application performance, however, is an open challenge. Ever increasing wire-delay relative to switching speed and the exponential cost of circuit complexity make simply scaling up existing processor designs futile. In this paper, we present an alternative to superscalar design, WaveScalar. WaveScalar is a dataflow instruction set architecture and execute ...

### **14 The impact of architectural trends on operating system performance**

M. Rosenblum, E. Bugnion, S. A. Herrod, E. Witchel, A. Gupta

December 1995 **ACM SIGOPS Operating Systems Review , Proceedings of the fifteenth ACM symposium on Operating systems principles**, Volume 29 Issue 5

Full text available: [pdf\(2.03 MB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

### **15 Caches and Memory Systems: Heterogeneous memory management for embedded systems**

Oren Avissar, Rajeev Barua, Dave Stewart

November 2001 **Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems**

Full text available: [pdf\(241.12 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper presents a technique for the efficient compiler management of software-exposed heterogeneous memory. In many lower-end embedded chips, often used in micro-controllers and DSP processors, heterogeneous memory units such as scratch-pad SRAM, internal DRAM, external DRAM and ROM are visible directly to the software, without automatic management by a hardware caching mechanism. Instead the memory units are mapped to different portions of the address space. Caches are avoided because of the ...

**Keywords:** embedded, heterogeneous, memory, storage

### **16 A simple graph-based intermediate representation**

Cliff Click, Michael Paleczny

March 1995 **ACM SIGPLAN Notices , Papers from the 1995 ACM SIGPLAN workshop on Intermediate representations**, Volume 30 Issue 3

Full text available: [pdf\(1.39 MB\)](#) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

We present a graph-based intermediate representation (IR) with simple semantics and a low-memory-cost C++ implementation. The IR uses a directed graph with labeled vertices and ordered inputs but unordered outputs. Vertices are labeled with opcodes, edges are unlabeled. We represent the CFG and basic blocks with the same vertex and edge structures. Each opcode is defined by a C++ class that encapsulates opcode-specific data and behavior. We use inheritance to abstract common opcode behavior ...

M. J. Smith, D. H. Sleeman  
June 1977 **ACM SIGART Bulletin**, Issue 63

Full text available: [pdf\(687.77 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#)

A general purpose Production System interpreter, APRIL, has been developed to model aspects of students problem solving in a CAI environment. APRIL incorporates most of the facilities common to earlier interpreters and also includes several additional features. One new feature allows "high-level" routines, such as those used by the interpreter itself in the matching and cycling phases, to be actions in production rules: hence it is possible for the user to "create" an interpreter which is suitable ...

#### **18 Trap-driven simulation with Tapeworm II**

Richard Uhlig, David Nagle, Trevor Mudge, Stuart Sechrist

November 1994 **Proceedings of the sixth international conference on Architectural support for programming languages and operating systems**, Volume 29, 28 Issue 11 , 5

Full text available: [pdf\(1.45 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Tapeworm II is a software-based simulation tool that evaluates the cache and TLB performance of multiple-task and operating system intensive workloads. Tapeworm resides in an OS kernel and causes a host machine's hardware to drive simulations with kernel traps instead of with address traces, as is conventionally done. This allows Tapeworm to quickly and accurately capture complete memory referencing behavior with a limited degradation in overall system performance. This paper compares trap- ...

**Keywords:** TLB, cache, memory system, trace-driven simulation, trap-driven simulation

#### **19 High-level architectural co-simulation using Esterel and C**

Andre Chatelain, Yves Mathys, Giovanni Placido, Alberto La Rosa, Luciano Lavagno

April 2001 **Proceedings of the ninth international symposium on Hardware/software codesign**

Full text available: [pdf\(711.50 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper introduces an architectural simulation environment, aimed at defining the best SOC architecture for complex system-level applications. The application is modeled using an abstract Timing Modeling Language, that describes the requests (e.g., memory accesses, I/Os, etc.) that the application makes to the architecture. The abstract architecture is modeled at the cycle-accurate level using a mixture of Esterel (a synchronous language) and C. We discuss the results of the application of ...

#### **20 A comparison of system monitoring methods, passive network monitoring and kernel instrumentation**

A. W. Moore, A. J. McGregor, J. W. Breen

January 1996 **ACM SIGOPS Operating Systems Review**, Volume 30 Issue 1

Full text available: [pdf\(1.89 MB\)](#) Additional Information: [full citation](#), [abstract](#), [index terms](#)

This paper presents the comparison of two methods of system monitoring, passive network monitoring and kernel instrumentation. The comparison is made on the basis of passive network monitoring being used as a replacement for kernel instrumentation in some situations. Despite the fact that the passive network monitoring technique is shown to perform poorly as a direct replacement for kernel instrumentation, this paper indicates the areas where passive network monitoring could be used to the greatest ...

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.  
[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)